COMP 410/539, "Software Construction Methodologies" Syllabus

Comp410/539 is a software engineering course where the all the students in the class work together to form a small software development "company" who is contracted to a "customer" to develop a large-scale project. The students are placed in a realistic development scenario that requires a state-of-the-art software solution. Typical projects involve multi-client/multi-server frameworks with database connectivity. Usually, the project is something that has never been attempted by anyone anywhere. The students must interact with a person who is their customer, gathering and developing specifications and requirements, setting timelines and giving progress reports. The students must create their own sub-teams, set milestones for all their members and work as a cohesive team on a very tight schedule. Comp410/539 uses a "pure discovery-based learning" approach where the students are placed into a difficult situation and only through diligent research, discussion, reflection, trials and errors will they be able to overcome the obstacles and in doing so, learn the critical software engineering skills they need to become successful in the real world.

Comp 410 is for undergraduate students while Comp 539 is for graduate students. All Comp 539 students are required to take on "evangelist" roles which cover project-wide cross-cutting concerns. The details of the evangelist roles will depend on the customer's needs.

Students are required to have a background that includes advanced object-oriented design and programming. In such, Comp310 for undergraduates or Comp404 for graduate students are required except by explicit permission by the instructor. Comp405/505 is also highly recommended. The class will typically use C# and .NET technologies, but those are not required as prerequisites.

Description

Traditional software engineering courses follow the Carnegie Mellon Software Engineering Institute's "Guidelines for Software Engineering Education Version 1" in a lecture based format that covers the suggested topics of life cycle, design, requirements and specifications, testing, etc. However, at Rice, the goal is to educate leaders in the field, so simply knowing the "book" practices is not enough. Rice students need to understand the *why* and the *how* of software engineering. The students need to understand what causes the problems that the various software

engineering methodologies address and what fundamentally one needs to do to correct them. This understanding is not tied to a particular methodology, but requires deep and careful reflection on real development processes and the issues that face them. The only way that students can truly understand a "real world" software development process is for them to be immersed in it themselves. They must experience all that goes wrong (and right) and figure out for themselves what it takes to correct the problems, overcome the hurdles and achieve their goals. They must also be acquainted with cutting edge technologies and more importantly, how to handle new technologies as part of the development process. To achieve this, Comp410 employs a non-traditional pedagogy based on "pure discovery learning". Instead of traditional lectures and homework assignments, the entire class works together as a single team, simulating the development department of a small software company faced with the daunting task of completing, in about 12 weeks, a very large project involving numerous unfamiliar and often "bleeding edge" technologies. (The first two weeks of the semester are taken up with including a warm-up project designed to help everyone get up to speed with some of the basic tools and technologies being used.) The students will do all the research, design, implementation and testing of their product on their own. Likewise, the students must conceive and execute their own notions of how to manage the project development. In addition, through discussion and weekly journals, the class reflects upon and attempts to solve the various issues and problems, both technical as well as project management, that arise during real-world multi-team software development. The roles of instructors and staff are as mentors and technical and project management resources, plus they provide feedback on student journal and milestone entries. Lectures will be eschewed except for specific technical topics as required. Discussion and debate is the preferred mode of learning.

Through "learning by doing" the class covers the following topics and more:

- State-of-the-art object oriented design and programming
    - Highly abstract, highly modular systems
    - Highly decoupled systems
    - Component-framework systems
- Project management
    - Team organization
    - Project and task specification
    - Team and individual task assignment and management
    - Human resource management
    - Milestone setting and management
    - Agile development (e.g. eXtreme Programming) vs. traditional up-front planning
    - Bug tracking and resolution
    - Deadline and crisis management
    - Managing and incorporating new technologies
- Communications
    - Interpersonal and inter-team
    - Volatile, high speed vs. persistent slow speed communications
    - Long and short term documentation
- New technologies
    - New programming language (C#) under new operating framework (.NET)

- Distributed and peer-to-peer systems
- Cloud computing systems
- New hardware platforms, e.g. TabletPC, Wiimote, Wii BalanceBoards, Kinect motion sensors, Wunderboard Internet-connected sensors, etc.
- New user interface modalities: pen/stylus, inking, gestures, handwriting recognition

The Computer Science curriculum at Rice University emphasizes diversity in programming languages and computing environments. To provide students with a solid understanding of .NET technology, our software engineering course, Comp 410/539 "Software Construction Methodology", has been based exclusively on C# and .NET for over 10 years. The course leads students through the process of specifying, designing and implementing a large software project that requires advanced and intricate applications of .NET solutions.

Journals

A critical part of the Comp410/539 learning experience is the journal-keeping in which all students are required to participate. Typically these weekly journals are responses to a set of questions about the week's technical and project management process achievements, obstacles and issues. This reflection on what was done, why it was done and what one thinks can be done to remedy the problems in the future is one of the lynchpins of learning the art and practice of software development.

Please see the Journal Tips and Traps wiki for more information on creating journals.

Students also create bi-weekly milestone reports to document their progress on the milestones that were set out for them over that time period.

Recommended Texts

- Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, *Design Patterns, Elements of Reusable Object-Oriented Software*, Addison Wesley, 1995. ISBN 0201633612. The bible of design patterns, a must-have for anyone serious about object oriented programming.
- Martin Fowler and Kendal Scott, *UML Distilled: A Brief Guide to the Standard Object Modeling Language, 2nd ed.*, Addison Wesley, 1999. ISBN 020165783X. Useful for learning how to fully use UML diagrams.
- Kent Beck, *eXtreme Programming eXplained, Embrace Change* , Addison-Wesley, ISBN: 0201616416. The definitive reference by the developer of XP.
- Martin Fowler,Kent Beck, John Brant,William Opdyke, Don Roberts, *Refactoring: Improving the Design of Existing Code*, Addison-Wesley, ISBN: 0201485672. The definitive book on refactoring.

Attendance

Class typically meets in the Oshman Engineering Design Kitchen.

100% class attendance is absolutely mandatory!

Regular participation in student-run meeting sessions is also expected.

Assignments

As this is a project-based course, the class grade will be determined from a combination of bi-weekly project milestones, weekly journals and the final product.

- **Warm-up Project**
  -
- **Weekly journals**
  - Each student will turn in a journal every Friday, whether or not class meets.
  - Journals must cover a specified set of topics and will be graded primarily on completeness and clarity.
- **Project milestones**
  - Each student will be responsible for at least one milestone that comes due every two weeks.
  - A score of 100% will be awarded for a fully completed milestone. Incomplete milestones will be assigned a score commensurate with the level of completeness and the importance of that milestone.
- **Peer Reviews**
  - Each student is expected to submit peer reviews of at least 3 journals from other students from the latest journal submissions.
- **Final product**
  - At the end of the semester, the final product will be assigned a score that will be applied equally to all students. The score will be based on the level of completeness of the product with regards to the contracted goals, the quality of the documentation, the quality of the design and the quality of its operation (i.e. how well it runs).

Additional Assignment and Course Information:

Add your contact information to the "Class Contacts" list. This list is NOT publicly viewable!

Pre-assignment: Read at least 2 years of previous journals and complete the associated survey. Fall 2011 is recommended since it is one of years with a large class size, meaning that their experiences might be more closely aligned with this year's.

Recommended: Read (and add to) the Quotes of Note!

Journals:

See the **Journal Tips and Traps**

Weekly journals will be due at 11:59 PM every Tues. (Group A) and Friday (Group B). Each student must complete their own journal. There is a 10 point/day (rounded up to nearest 5 pt/12 hrs) penalty for late journals. Note that that *each* journal represents approximately *5%* of the total class grade!

If a journal hasn't been made yet, make it yourself using the "**`Weekly journal with peer review`**" template. See the **Journal Tips and Traps** for instructions.

Journal Peer Reviews:

Group A journals are peer-reviewed by Group B and vice versa. Peer reviews are due within 24 hours of the deadline for the journal submission.

Each student must peer-review at least 3 journals (only enter scores for 3 journals max).

Peer reviewing consists of:

- Adding comments to the existing journal covering but not limited to
  - Clarity and completeness of the submission -- explicitly point out where the issues occur and why you think they are vague and/or incomplete.
  - Suggestions for technical or development process improvement.
  - Additional thoughts or comments on the topics being discussed.
- When adding comments, use any color **EXCEPT** BRIGHT RED or DARK RED which for the staff (preferably an easy-to-read color, e.g. brown, dark green, blue, etc.), and preface all comments with your NetID, e.g. "netid: Good point! But what if..."
- The staff will also be adding comments, both in the individual journals as well as in a the General Journal Feedback page. Typically, this will be done AFTER the peer-reviews are complete.
- Assigning a score to the journal from 0-100%. An approximate grading guideline is
  - 95%-100% -- Essentially a perfect journal that is clear and complete in all aspects save one or two minor issues.
  - 90% - 95% -- One major or a few minor issues.
  - 85% - 90% -- Two or three major issues
  - 80% - 85% -- Four to six major issues.
  - 70% - 80% -- Major parts of the journal are missing but here are some parts are ok.
  - 30% - 70% -- Major parts of the journal are missing and there are major problems with what is there.
  - 0% - 30% -- Journal is essentially non-existent but credit given for doing something at all.
- The staff will be adding their own score and a final score, which not necessarily an equal weighting between peer and staff scores, will be calculated.

Peer reviews DO count towards the final course grade.

Milestone Reports

During the Final Project development process, Milestone Reports will be due approximately every other week, corresponding to a milestone tasking period. These reports (survey submissions, actually) are individual descriptions of exactly what parts of the assigned milestone tasks were and were not completed over the covered time period. Note that the ability to write a milestone report is predicated on the ability to articulate one assigned tasks!

Direct links to tasking documentation is encouraged.

Software

The following software is required for Comp410. Ask the staff how to get it if you don't already have it.

- Visual Studio -- available for free from the staff. The freeware versions of Visual Studio are not sufficient for Comp410/539!
- Windows (latest version) -- this is preferred but not mandatory. This is available for free from the staff.
  - If your Mac is powerful enough, install Windows as a dual boot or as a virtual machine.

- SDK's
    - Microsoft Azure Cloud
    - Others as needed.
- Team Foundation Server
    - TFS Connection Info (login req'd)

Grading (subject to change!):

- Warm-up Project: 10%
- Journals: 45%
- Peer reviews: 10%
- Milestone Reports: 10%
- Final Deliverables: 25%

Teaching Staff

Instructors:

- Dr. Stephen Wong

Teaching Assistants:

- TBA

ADA Statement: If you have a documented disability that will impact your work in this class, please contact me to discuss your needs. Additionally, you will need to register with the Disability Support Services Office in the Allen Center.